DIRECTED ORDERED ACYCLIC GRAPHS

ASYMPTOTIC ANALYSIS AND EFFICIENT RANDOM SAMPLING

Martin Pépin
joint work with Antoine Genitrini & Alfredo Viola

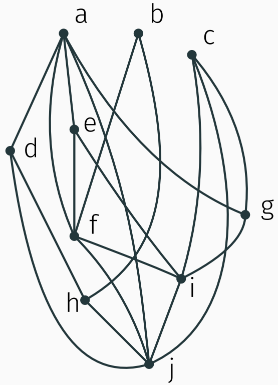
March 14, 2023
Journées ALEA 2023

LIPN

Directed Acyclic Graphs

Directed Acyclic Graph (DAG)

- A finite set of vertices V e.g. $\{a, b, c, \ldots, j\}$;
- a set of directed edges $E \subseteq V \times V$;
- no cycles.

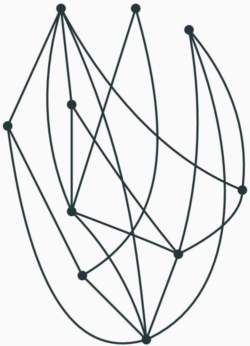


Directed Acyclic Graphs

Directed Acyclic Graph (DAG)

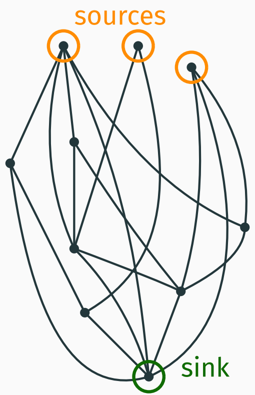
- A finite set of vertices V e.g. $\{a, b, c, \ldots, j\}$;
- a set of directed edges $E \subseteq V \times V$;
- no cycles.

Without labels: **Unlabelled DAGs**



Directed Acyclic Graph (DAG)

- A finite set of vertices V e.g. $\{a, b, c, \ldots, j\}$;
- a set of directed edges $E \subseteq V \times V$;
- no cycles.



Without labels: **Unlabelled DAGs**

Why DAGs?

Omnipresent data structure:

- Encoding partial orders in scheduling problems;
- Git histories;
- genealogy trees (those are not trees!);
- compacted trees (or XML documents, etc.);
- hash-consing...

Why DAGs?

Omnipresent data structure:

- **Encoding partial orders in scheduling problems;**
- Git histories;
- genealogy trees (those are not trees!);
- compacted trees (or XML documents, etc.);
- hash-consing...

Why DAGs?

Omnipresent data structure:

- Encoding partial orders in scheduling problems;
- Git histories;
- genealogy trees (those are not trees!);
- **compacted trees (or XML documents, etc.);**
- **hash-consing...**

State of the art

Labelled DAGs:

- Counting by vertices: [Rob70; Rob73; Sta73]

State of the art

Labelled DAGs:

- Counting by vertices: [Rob70; Rob73; Sta73]
- Counting by vertices and edges: [Ges96]

State of the art

Labelled DAGs:

- Counting by vertices: [Rob70; Rob73; Sta73]
- Counting by vertices and edges: [Ges96]
- Uniform sampling: [MDB01], [KM15]

State of the art

Labelled DAGs:

- Counting by vertices: [Rob70; Rob73; Sta73]
- Counting by vertices and edges: [Ges96]
- Uniform sampling: [MDB01], [KM15]

Unlabelled DAGs:

- Counting by vertices: [Rob70; Rob77]

State of the art

Labelled DAGs:

- Counting by vertices: [Rob70; Rob73; Sta73]
- Counting by vertices and edges: [Ges96]
- Uniform sampling: [MDB01], [KM15]

Unlabelled DAGs:

- Counting by vertices: [Rob70; Rob77]

Compacted trees:

- Counting in the binary case: [GGKW20; EFW21]

State of the art

Labelled DAGs:

- Counting by vertices: [Rob70; Rob73; Sta73]
- Counting by vertices and edges: [Ges96] •
- Uniform sampling: [MDB01], [KM15]

Unlabelled DAGs:

- Counting by vertices: [Rob70; Rob77]

Compacted trees:

- Counting in the binary case: [GGKW20; EFW21]

Problems:

- Inclusion-exclusion

State of the art

Labelled DAGs:

- Counting by vertices: [Rob70; Rob73; Sta73]
- Counting by vertices and edges: [Ges96] ●
- Uniform sampling: [MDB01], [KM15] ●

Unlabelled DAGs:

- Counting by vertices: [Rob70; Rob77]

Compacted trees:

- Counting in the binary case: [GGKW20; EFW21]

Problems:

- ● Inclusion-exclusion
- ● No or little control over the number of edges

State of the art

Labelled DAGs:

- Counting by vertices: [Rob70; Rob73; Sta73]
- Counting by vertices and edges: [Ges96] ●
- Uniform sampling: [MDB01], [KM15] ●

Unlabelled DAGs:

- Counting by vertices: [Rob70; Rob77]

Compacted trees:

- Counting in the binary case: [GGKW20; EFW21] ●

Problems:

- ● Inclusion-exclusion
- ● No or little control over the number of edges
- ● Only binary

Outline of the presentation

Background

Directed ordered acyclic graphs

↳ *definition and recursive decomposition*

Asymptotic analysis

↳ *matrix encoding*
↳ *asymptotic result*
↳ *faster sampler*

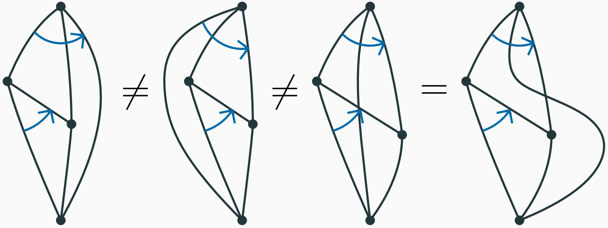
Bonus: labelled DAGs

↳ *another way of counting*

A new kind of DAG

Directed Ordered Acyclic Graphs

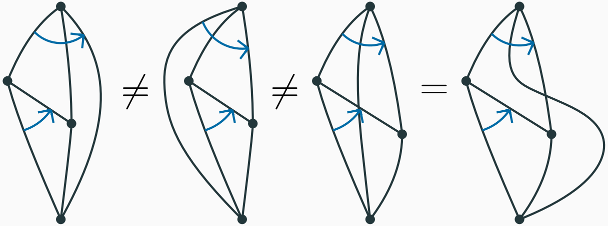
DOAG = Unlabelled DAG
+ a total order on the **outgoing** edges of each vertex
+ a total order on the sources

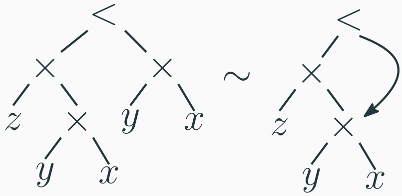


A new kind of DAG

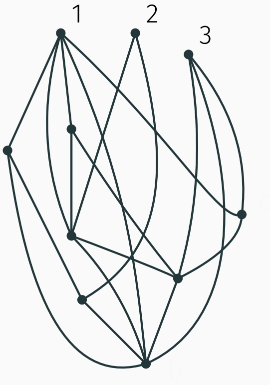
Directed Ordered Acyclic Graphs

DOAG = Unlabelled DAG
+ a total order on the **outgoing** edges of each vertex
+ a total order on the sources



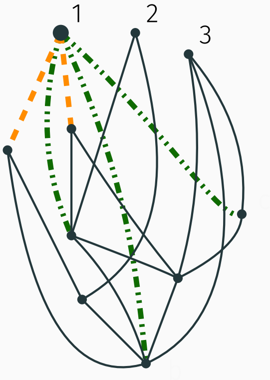


Recursive decomposition



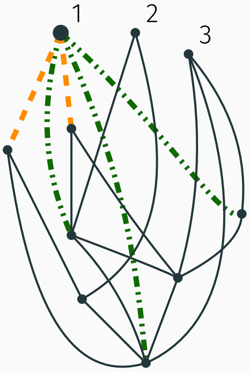
n vertices, m edges, k sources

Recursive decomposition

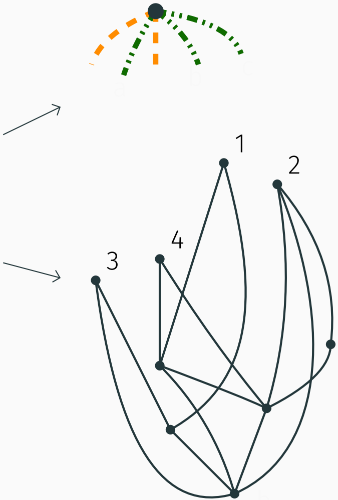


n vertices, m edges, k sources

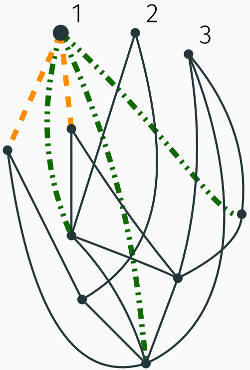
Recursive decomposition



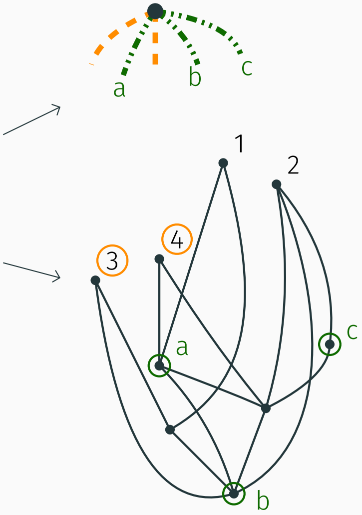
n vertices, m edges, k sources



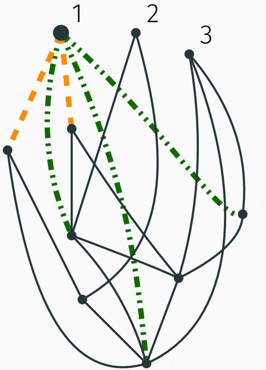
Recursive decomposition



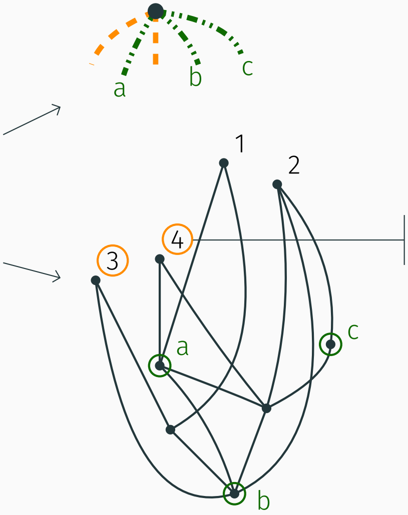
n vertices, m edges, k sources



Recursive decomposition

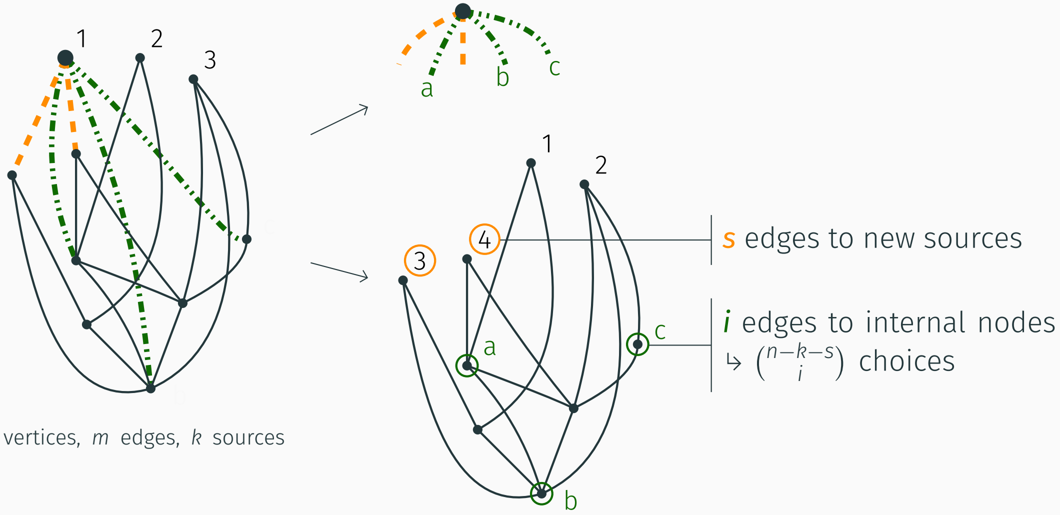


n vertices, m edges, k sources



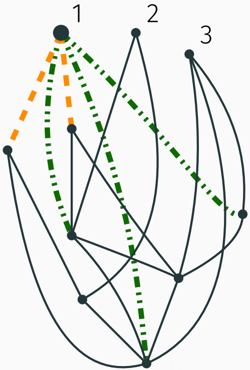
s edges to new sources

Recursive decomposition

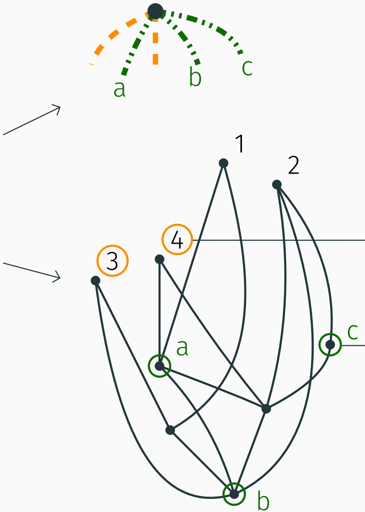


n vertices, m edges, k sources

Recursive decomposition



n vertices, m edges, k sources



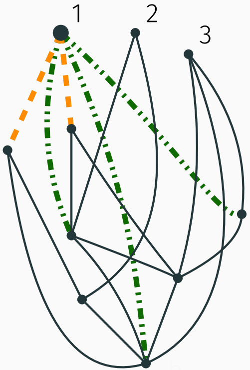
s edges to new sources

i edges to internal nodes

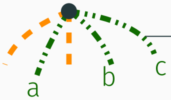
↳ $\binom{n-k-s}{i}$ choices

$(n-1)$ vertices, $(m-i-s)$ edges, $(k+s-1)$ sources

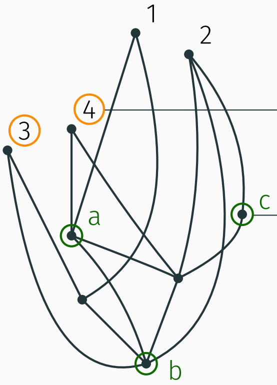
Recursive decomposition



n vertices, m edges, k sources



permute the […] edges
interleave […] with […]
↳ $i!\binom{i+s}{i}$ choices



s edges to new sources

i edges to internal nodes
↳ $\binom{n-k-s}{i}$ choices

$(n-1)$ vertices, $(m-i-s)$ edges, $(k+s-1)$ sources

Recurrence formula

Counting formula

$$D_{n,m,k} = \#\{\text{DOAGs with } n \text{ vertices, } m \text{ edges and } k \text{ sources}\}$$
$$= \sum_{i,s \geq 0} D_{n-1,m-i-s,k+s-1} \binom{n-k-s}{i} i! \binom{i+s}{i}$$

Recurrence formula

Counting formula

$$D_{n,m,k} = \#\{\textit{DOAGs with } n \textit{ vertices, } m \textit{ edges and } k \textit{ sources}\}$$
$$= \sum_{i,s\geq 0} D_{n-1,m-i-s,k+s-1} \binom{n-k-s}{i} i! \binom{i+s}{i}$$

Counting: up to $n, k = N$ and $m = M$:

→ $O(N^4 M)$ operations;

→ on integers of bit-size $O(M \log M)$.

Recurrence formula

Counting formula

$$D_{n,m,k} = \#\{\textit{DOAGs with } n \textit{ vertices, } m \textit{ edges and } k \textit{ sources}\}$$
$$= \sum_{i,s \geq 0} D_{n-1,m-i-s,k+s-1} \binom{n-k-s}{i} i! \binom{i+s}{i}$$

Counting: up to $n, k = N$ and $m = M$:

→ $O(N^4 M)$ operations;

→ on integers of bit-size $O(M \log M)$.

Random sampling = counting: $O\left(\sum_{v \text{ vertex}} d_v^2\right) = O(\min(N^3, M^2))$.

↳ out-degree of v

Recurrence formula

Counting formula

$$D_{n,m,k} = \#\{\textit{DOAGs with } n \textit{ vertices, } m \textit{ edges and } k \textit{ sources}\}$$
$$= \sum_{i,s \geq 0} D_{n-1,m-i-s,k+s-1} \binom{n-k-s}{i} i! \binom{i+s}{i}$$

Counting: up to $n, k = N$ and $m = M$:
→ $O(N^4 M)$ operations;
→ on integers of bit-size $O(M \log M)$.

Random sampling = gnitnuoɔ: $O\left(\sum_{v \text{ vertex}} d_v^2\right) = O(\min(N^3, M^2))$.
↳ out-degree of v

In practice: for $M \approx 400$, one sink: → counting = several minutes
→ sampling = a few ms

Outline of the presentation

Background

Directed ordered acyclic graphs

↳ *definition and recursive decomposition*

Asymptotic analysis

↳ *matrix encoding*
↳ *asymptotic result*
↳ *faster sampler*

Bonus: labelled DAGs

↳ *another way of counting*

A first asymptotic result

Asymptotics: (approximately) how many large DOAGs are there when $n, m \to \infty$? (And what about k?)

A first asymptotic result

Asymptotics: (approximately) how many large DOAGs are there when $n, m \to \infty$? (And what about k?)

Simplification: drop parameters, only count by vertices.

$$D_n \stackrel{\text{def}}{=} \#\{\text{DOAGs with } n \text{ vertices, one source}\}$$

A first asymptotic result

Asymptotics: (approximately) how many large DOAGs are there when $n, m \to \infty$? (And what about k?)

Simplification: drop parameters, only count by vertices.

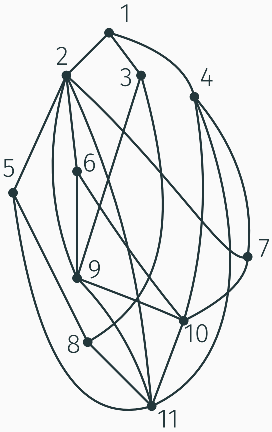
$$D_n \stackrel{\text{def}}{=} \#\{\text{DOAGs with } n \text{ vertices, one source}\}$$

Number of single-source DOAGs (P., Viola, 2023+)

$$D_n \underset{n\to\infty}{\sim} c \cdot n^{-1/2} \cdot e^{n-1} \cdot ¡n-1!$$

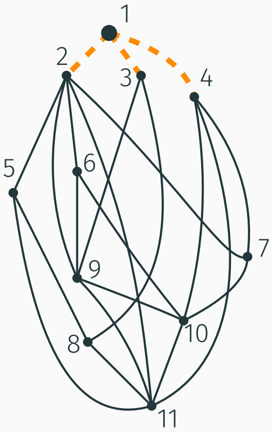
for $c \approx 0.4967$ *and where* $¡x! = \prod_{k=1}^{x} k!$.

Matrix encoding



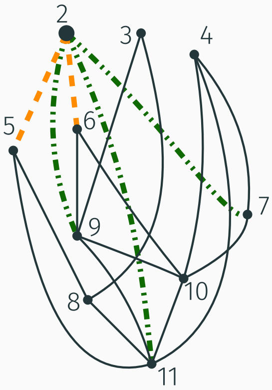
1	2	3	4	5	6	7	8	9	10	11	
											1
											2
											3
											4
											5
											6
											7
											8
											9
											10
											11

Matrix encoding



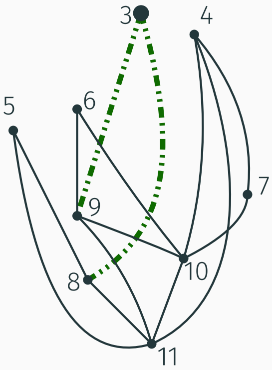
	1	2	3	4	5	6	7	8	9	10	11
1		**1**	**2**	**3**							
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											

Matrix encoding



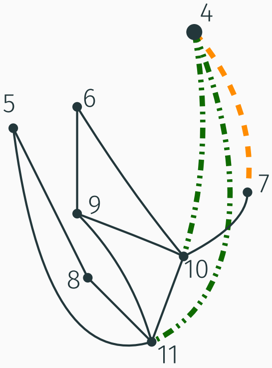
1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
											3
											4
											5
											6
											7
											8
											9
											10
											11

Matrix encoding



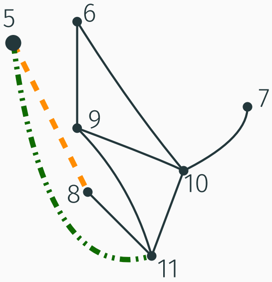
1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
							2	**1**			3
											4
											5
											6
											7
											8
											9
											10
											11

Matrix encoding



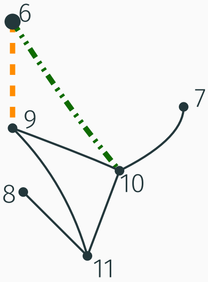
	1	2	3	4	5	6	7	8	9	10	11
1		**1**	**2**	**3**							
2					**1**	**3**	**5**		**2**		**4**
3								**2**	**1**		
4							**3**			**1**	**2**
5											
6											
7											
8											
9											
10											
11											

Matrix encoding



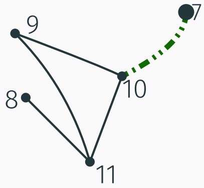
	1	2	3	4	5	6	7	8	9	10	11
1		**1**	**2**	**3**							
2					**1**	**3**	**5**		**2**		**4**
3								**2**	**1**		
4							**3**			**1**	**2**
5								**2**			**1**
6											
7											
8											
9											
10											
11											

Matrix encoding



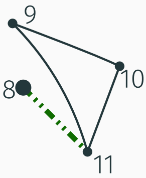
	1	2	3	4	5	6	7	8	9	10	11
1		**1**	**2**	**3**							
2					**1**	**3**	**5**		**2**		**4**
3								**2**	**1**		
4							**3**			**1**	**2**
5								**2**			**1**
6									**1**	**2**	
7											
8											
9											
10											
11											

Matrix encoding



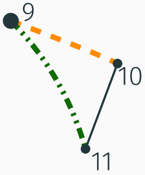
1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
							2	**1**			3
						3			**1**	**2**	4
							2			**1**	5
								1	**2**		6
									1		7
											8
											9
											10
											11

Matrix encoding



1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
							2	**1**			3
						3			**1**	**2**	4
							2			**1**	5
								1	**2**		6
									1		7
										1	8
											9
											10
											11

Matrix encoding



1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
							2	**1**			3
						3			**1**	**2**	4
							2			**1**	5
								1	**2**		6
									1		7
										1	8
									2	**1**	9
											10
											11

Matrix encoding

	1	2	3	4	5	6	7	8	9	10	11
1		**1**	**2**	**3**							
2					**1**	**3**	**5**		**2**		**4**
3								**2**	**1**		
4							**3**			**1**	**2**
5								**2**			**1**
6									**1**	**2**	
7										**1**	
8											**1**
9										**2**	**1**
10											**1**
11											

Matrix encoding

1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
							2	**1**			3
						3			**1**	**2**	4
							2			**1**	5
								1	**2**		6
									1		7
										1	8
									2	**1**	9
										1	10
											11

Matrix encoding

1. strict upper triangular matrix;

1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
							2	**1**			3
						3			**1**	**2**	4
							2			**1**	5
								1	**2**		6
									1		7
										1	8
									2	**1**	9
										1	10
											11

1. strict upper triangular matrix;
2. lines use an interval of values;

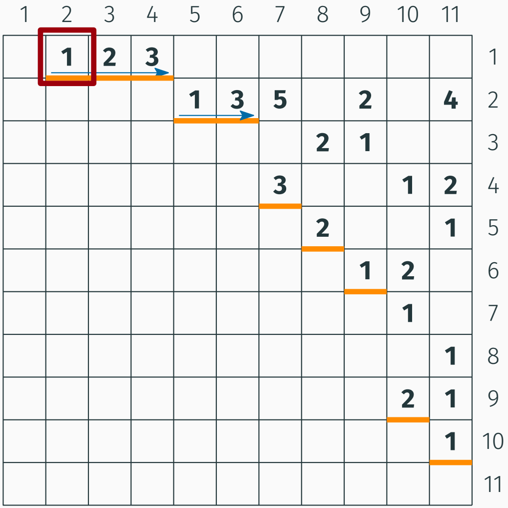
1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
							2	**1**			3
						3			**1**	**2**	4
							2			**1**	5
								1	**2**		6
									1		7
										1	8
									2	**1**	9
										1	10
											11

Matrix encoding

1. strict upper triangular matrix;
2. lines use an interval of values;
3. $a_{1,2} \neq 0$;

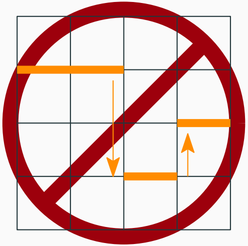
1	2	3	4	5	6	7	8	9	10	11	
	1	**2**	**3**								1
				1	**3**	**5**		**2**		**4**	2
							2	**1**			3
						3			**1**	**2**	4
							2			**1**	5
								1	**2**		6
									1		7
										1	8
									2	**1**	9
										1	10
											11

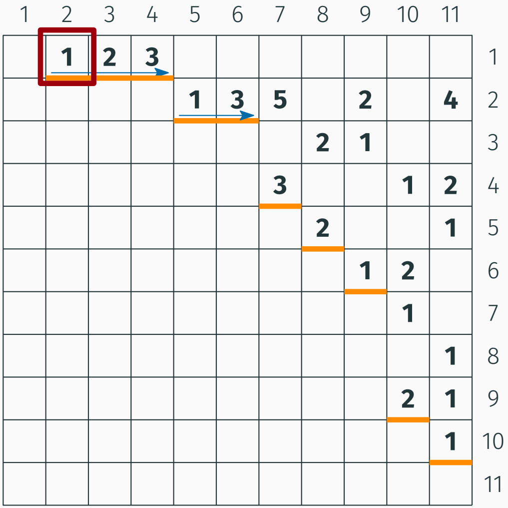
1. strict upper triangular matrix;
2. lines use an interval of values;
3. $a_{1,2} \neq 0$;
4. increasing numbers above orange lines;



Matrix encoding

1. strict upper triangular matrix;
2. lines use an interval of values;
3. $a_{1,2} \neq 0$;
4. increasing numbers above orange lines;
5. orange lines go down.





Proof sketch (1/3)

The plan: 1. Upper bound 2. Lower bound 3. Bootstrapping

Proof sketch (1/3)

The plan: **1. Upper bound** 2. Lower bound 3. Bootstrapping

Proof sketch (1/3)

The plan: **1. Upper bound** 2. Lower bound 3. Bootstrapping

6	**1**		**5**		**2**	**4**		**3**	$=$	**6**	**1**	**5**	**2**	**4**	**3**	$\sqcup\!\sqcup$			

Proof sketch (1/3)

The plan: **1. Upper bound** 2. Lower bound 3. Bootstrapping

6	**1**		**5**		**2**	**4**		**3**	$=$	**6**	**1**	**5**	**2**	**4**	**3**	$\shuffle$			

$$\text{Variation} = \textsc{Seq}(\mathcal{Z}) \star \textsc{Set}(\mathcal{Z})$$

Proof sketch (1/3)

The plan: **1. Upper bound** 2. Lower bound 3. Bootstrapping

6	1		5		2	4		3

$=$

6	1	5	2	4	3

$\sqcup\!\sqcup$

$$\text{Variation} = \textsc{Seq}(\mathcal{Z}) \star \textsc{Set}(\mathcal{Z})$$

$$V(z) = (1-z)^{-1} e^{z}$$

Proof sketch (1/3)

The plan: **1. Upper bound** 2. Lower bound 3. Bootstrapping

6	1		5		2	4		3	=	6	1	5	2	4	3	⧢				

$$\text{Variation} = \textsc{Seq}(\mathcal{Z}) \star \textsc{Set}(\mathcal{Z})$$

$$V(z) = (1-z)^{-1} e^{z}$$

$$v_n = e \cdot n! - o(1)$$

Proof sketch (1/3)

The plan: **1. Upper bound** 2. Lower bound 3. Bootstrapping

6	1		5		2	4		3

$=$

6	1	5	2	4	3

$\sqcup\!\sqcup$

$$\text{Variation} = \textsc{Seq}(\mathcal{Z}) \star \textsc{Set}(\mathcal{Z})$$

$$V(z) = (1-z)^{-1}e^{z}$$

$$v_n = e \cdot n! - o(1)$$

$\#\{\text{DOAG matrices}\} = \#\{\text{collections of rows}\} \leq \#\{\text{collections of variations}\}$

Proof sketch (1/3)

The plan: **1. Upper bound** 2. Lower bound 3. Bootstrapping

6	1		5		2	4		3

$=$

6	1	5	2	4	3

$\sqcup\!\sqcup$

$$\text{Variation} = \textsc{Seq}(\mathcal{Z}) \star \textsc{Set}(\mathcal{Z})$$

$$V(z) = (1-z)^{-1}e^{z}$$

$$v_n = e \cdot n! - o(1)$$

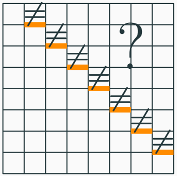
$\#\{\text{DOAG matrices}\} = \#\{\text{collections of rows}\} \leq \#\{\text{collections of variations}\}$

$$D_n \leq \prod_{k=1}^{n-1} v_k \leq e^{n-1}\ ¡n-1!$$

Proof sketch (2/3)

The plan: 1. Upper bound **2. Lower bound** 3. Bootstrapping

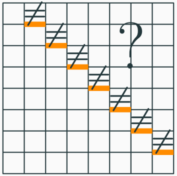
$\{$DOAG matrices$\} \supseteq$



(constraints are automatically satisfied)

Proof sketch (2/3)

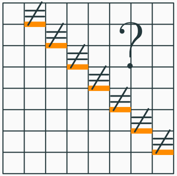
The plan: 1. Upper bound **2. Lower bound** 3. Bootstrapping

{DOAG matrices} ⊇  (constraints are automatically satisfied)

$\#\,\boxed{\neq}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?} = V_k - V_{k-1}$

Proof sketch (2/3)

The plan: 1. Upper bound **2. Lower bound** 3. Bootstrapping

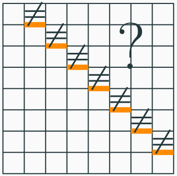
{DOAG matrices} $\supseteq$ 

(constraints are automatically satisfied)

$$\#\,\boxed{\neq}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?} = v_k - v_{k-1} = e \cdot k! \cdot \left(1 - \frac{1}{k} - o\left(\frac{1}{(k-1)!}\right)\right)$$

Proof sketch (2/3)

The plan:	1. Upper bound	**2. Lower bound**	3. Bootstrapping

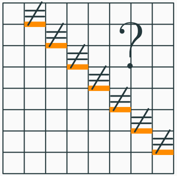
{DOAG matrices} $\supseteq$  (constraints are automatically satisfied)

$$\#\,\boxed{\neq}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?} = v_k - v_{k-1} = e \cdot k! \cdot \left(1 - \frac{1}{k} - o\left(\frac{1}{(k-1)!}\right)\right)$$

$$D_n \geq e^{n-1}\,\text{¡}n-1! \prod_{k=2}^{n-1} \left(\frac{k-1}{k} + o\left(\frac{1}{(k-1)!}\right)\right)$$

Proof sketch (2/3)

The plan: 1. Upper bound **2. Lower bound** 3. Bootstrapping

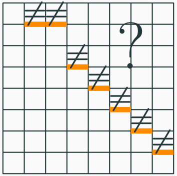
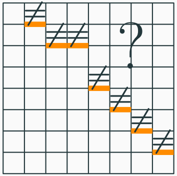
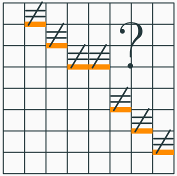
{DOAG matrices} $\supseteq$  (constraints are automatically satisfied)

$$\#\boxed{\neq}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?} = v_k - v_{k-1} = e \cdot k! \cdot \left(1 - \frac{1}{k} - o\left(\frac{1}{(k-1)!}\right)\right)$$

$$D_n \geq e^{n-1}\ ¡n-1! \prod_{k=2}^{n-1}\left(\frac{k-1}{k} + o\left(\frac{1}{(k-1)!}\right)\right) \geq e^{n-1}\ ¡n-1!\ \frac{A}{n} \quad \text{for some } A > 0$$

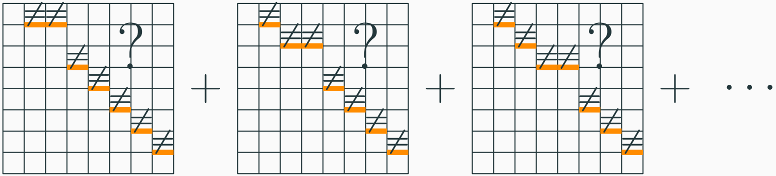
Proof sketch (2'/3)

The plan: 1. Upper bound **2'. Better lower bound** 3. Bootstrapping

{DOAG matrices} $\supseteq$  $+$  $+$  $+ \cdots$

Proof sketch (2’/3)

The plan: 1. Upper bound **2’. Better lower bound** 3. Bootstrapping

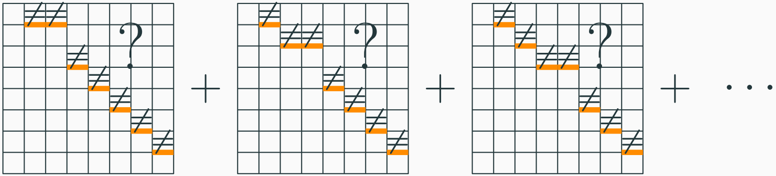
{DOAG matrices} ⊇ 

$$D_n \geq \frac{A' \cdot \ln(n)}{n} e^{n-1} \text{¡} n - 1 \text{!}$$

Proof sketch (2'/3)

The plan: 1. Upper bound **2'. Better lower bound** 3. Bootstrapping

{DOAG matrices} $\supseteq$

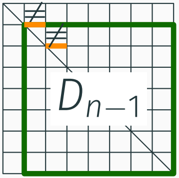


$$D_n \geq \frac{A' \cdot \ln(n)}{n}\, e^{n-1}\, \text{¡}n - 1!$$

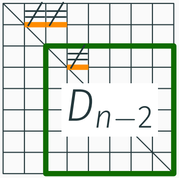
$$P_n = \frac{D_n}{e^{n-1}\, \text{¡}n - 1!} \quad \Rightarrow \quad \frac{A' \cdot \ln(n)}{n} \leq P_n \leq 1$$

The plan: 1. Upper bound 2. Better lower bound **3. Bootstrapping**

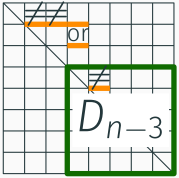
{DOAG matrices} =



+



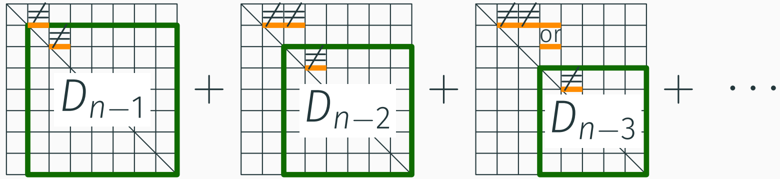
+



+ $\cdots$

Proof sketch (3/3)

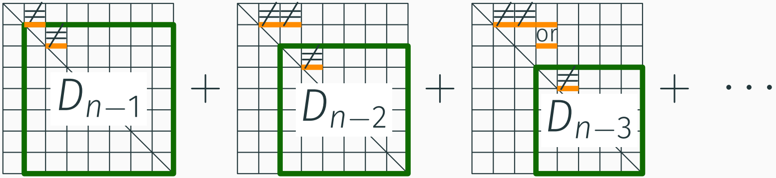
The plan: 1. Upper bound 2. Better lower bound **3. Bootstrapping**

{DOAG matrices} = 

$$D_n = (v_{n-1} - v_{n-2})D_{n-1} + \frac{1}{2}(v_{n-1} - 2v_{n-2} + v_{n-3})v_{n-3}D_{n-2} + \cdots$$

Proof sketch (3/3)

The plan: 1. Upper bound 2. Better lower bound **3. Bootstrapping**

{DOAG matrices} = 

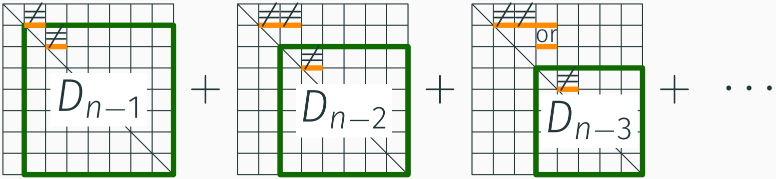
$$D_n = (v_{n-1} - v_{n-2})D_{n-1} + \frac{1}{2}(v_{n-1} - 2v_{n-2} + v_{n-3})v_{n-3}D_{n-2} + \cdots$$

$$P_n = \left(1 - \frac{1}{n-1}\right)P_{n-1} + \frac{1}{2(n-2)}\left(1 - \frac{2}{n-1} + \frac{1}{(n-1)(n-2)}\right)P_{n-2} + \cdots$$

Proof sketch (3/3)

The plan: 1. Upper bound 2. Better lower bound **3. Bootstrapping**

{DOAG matrices} =



$$D_n = (v_{n-1} - v_{n-2})D_{n-1} + \frac{1}{2}(v_{n-1} - 2v_{n-2} + v_{n-3})v_{n-3}D_{n-2} + \cdots$$

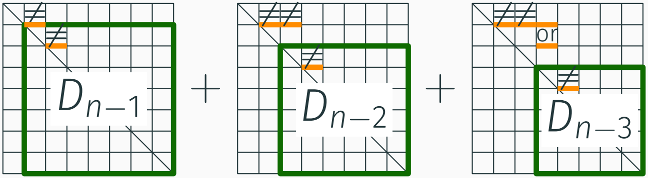
$$P_n = \left(1 - \frac{1}{n-1}\right) P_{n-1} + \frac{1}{2(n-2)} \left(1 - \frac{2}{n-1} + \frac{1}{(n-1)(n-2)}\right) P_{n-2} + \cdots$$

$$P_n \sim P_{n-1}$$

Proof sketch (3/3)

The plan: 1. Upper bound 2. Better lower bound **3. Bootstrapping**

{DOAG matrices} =



$+ \cdots$

$$D_n = (v_{n-1} - v_{n-2})D_{n-1} + \frac{1}{2}(v_{n-1} - 2v_{n-2} + v_{n-3})v_{n-3}D_{n-2} + \cdots$$

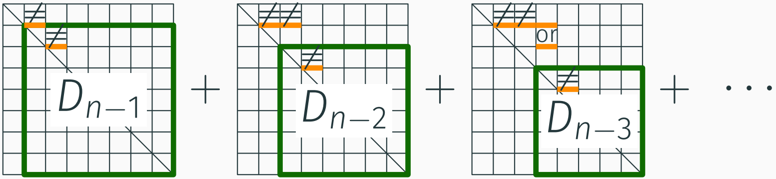
$$P_n = \left(1 - \frac{1}{n-1}\right)P_{n-1} + \frac{1}{2(n-2)}\left(1 - \frac{2}{n-1} + \frac{1}{(n-1)(n-2)}\right)P_{n-2} + \cdots$$

$$P_n \sim P_{n-1}$$

Proof sketch (3/3)

The plan: 1. Upper bound 2. Better lower bound **3. Bootstrapping**

{DOAG matrices} =



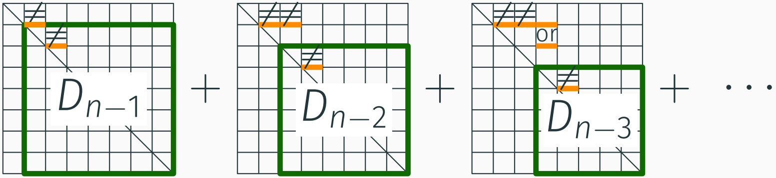
$$D_n = (v_{n-1} - v_{n-2})D_{n-1} + \frac{1}{2}(v_{n-1} - 2v_{n-2} + v_{n-3})v_{n-3}D_{n-2} + \cdots$$

$$P_n = \left(1 - \frac{1}{n-1}\right)P_{n-1} + \frac{1}{2(n-2)}\left(1 - \frac{2}{n-1} + \frac{1}{(n-1)(n-2)}\right)P_{n-2} + \cdots$$

$$P_n = P_{n-1}\left(1 - \frac{1}{2n} + O\left(n^{-2}\right)\right)$$

The plan: 1. Upper bound 2. Better lower bound **3. Bootstrapping**

{DOAG matrices} =



$$D_n = (v_{n-1} - v_{n-2})D_{n-1} + \frac{1}{2}(v_{n-1} - 2v_{n-2} + v_{n-3})v_{n-3}D_{n-2} + \cdots$$

$$P_n = \left(1 - \frac{1}{n-1}\right)P_{n-1} + \frac{1}{2(n-2)}\left(1 - \frac{2}{n-1} + \frac{1}{(n-1)(n-2)}\right)P_{n-2} + \cdots$$

$$P_n = P_{n-1}\left(1 - \frac{1}{2n} + O\left(n^{-2}\right)\right) \Rightarrow \boxed{P_n \sim c \cdot n^{-1/2}}$$

Random sampling again!

Corollary

$$\frac{D_n}{\#\{\text{matrices of variations of sizes } 1, 2, \ldots, n-1\}} \sim c \cdot n^{-\frac{1}{2}}$$

Random sampling again!

Corollary

$$\frac{D_n}{\#\{\textit{matrices of variations of sizes } 1, 2, \ldots, n-1\}} \sim c \cdot n^{-\frac{1}{2}}$$

: draw variation matrices until they correspond to a DOAG

Random sampling again!

Corollary

$$\frac{D_n}{\#\{\textit{matrices of variations of sizes } 1, 2, \ldots, n-1\}} \sim c \cdot n^{-\frac{1}{2}}$$

: draw variation matrices until they correspond to a DOAG

(Naive) complexity: #rejections $\times$ Cost(one generation)

Random sampling again!

Corollary

$$\frac{D_n}{\#\{\textit{matrices of variations of sizes } 1, 2, \ldots, n-1\}} \sim c \cdot n^{-\frac{1}{2}}$$

: draw variation matrices until they correspond to a DOAG

(Naive) complexity: $\#\text{rejections} \times \mathsf{Cost}(\text{one generation})$

Generating one variation: $\sim n \log_2(n)$ random bits.

Random sampling again!

Corollary

$$\frac{D_n}{\#\{\textit{matrices of variations of sizes } 1,2,\ldots,n-1\}} \sim c \cdot n^{-\frac{1}{2}}$$

: draw variation matrices until they correspond to a DOAG

(Naive) complexity: $O(\sqrt{n} \cdot n^2 \ln(n))$ random bits

Generating one variation: $\sim n \log_2(n)$ random bits.

Random sampling again!

Corollary

$$\frac{D_n}{\#\{\textit{matrices of variations of sizes } 1, 2, \ldots, n-1\}} \sim c \cdot n^{-\frac{1}{2}}$$

: draw variation matrices until they correspond to a DOAG

(Naive) complexity: $O(\sqrt{n} \cdot n^2 \ln(n))$ random bits

Generating one variation: $\sim n \log_2(n)$ random bits.

Better complexity:

$$\begin{aligned} &\text{Cost}(\text{one full generation}) + \#\text{rejections} \times \text{Cost}(\text{one failed generation}) \\ &= \frac{n^2}{2} \log_2(n) + O\left(\sqrt{n} \cdot \mathbf{Cost(}\text{one failed generation}\mathbf{)}\right) \end{aligned}$$

Random sampling again!

Corollary

$$\frac{D_n}{\#\{\textit{matrices of variations of sizes } 1, 2, \ldots, n-1\}} \sim c \cdot n^{-\frac{1}{2}}$$

: draw variation matrices until they correspond to a DOAG

(Naive) complexity: $O(\sqrt{n} \cdot n^2 \ln(n))$ random bits

Generating one variation: $\sim n \log_2(n)$ random bits.

Better complexity:

$$\begin{aligned} &\mathrm{Cost}(\text{one full generation}) + \#\text{rejections} \times \mathrm{Cost}(\text{one failed generation}) \\ &= \frac{n^2}{2} \log_2(n) + O\left(\sqrt{n} \cdot \mathbf{n \ln(n)}\right) \end{aligned}$$

Perspectives

- Study the shape of big DOAGs

Perspectives

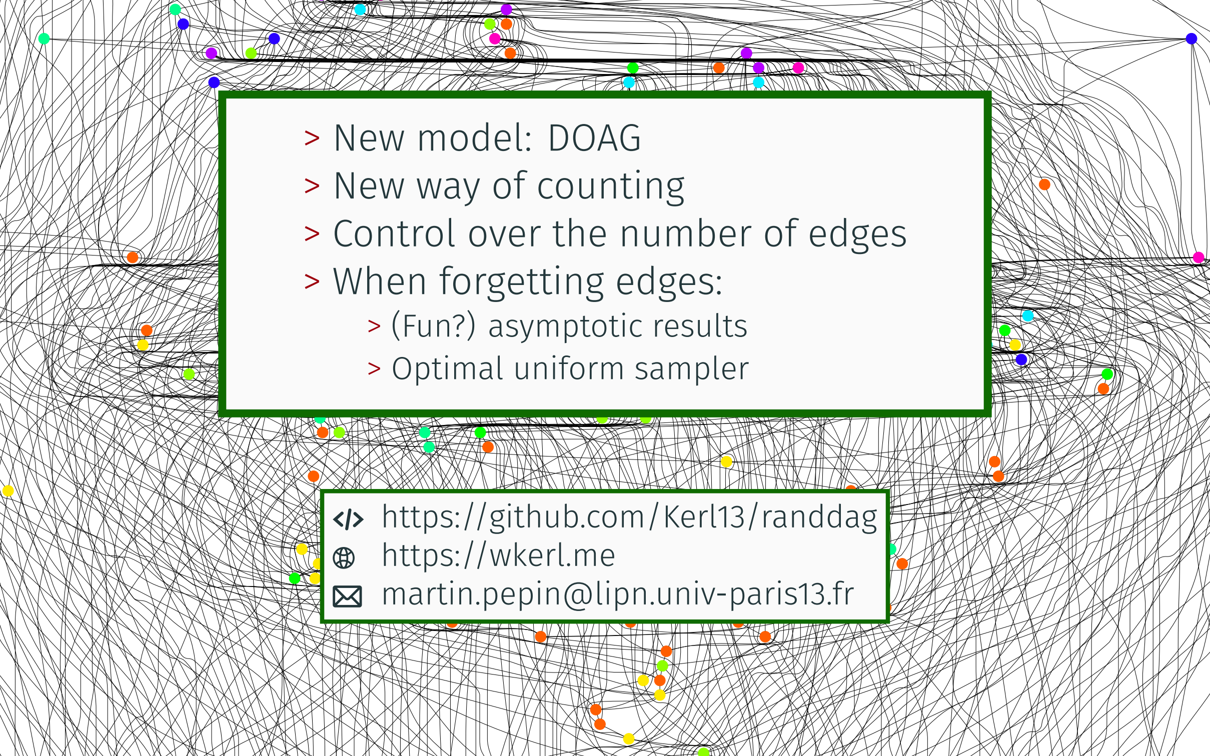
- Study the shape of big DOAGs
- Do something similar for labelled DAGs
    - Collaboration with Philippe Marchal

Perspectives

- Study the shape of big DOAGs
- Do something similar for labelled DAGs
    - Collaboration with Philippe Marchal
- Multigraph equivalent: DOAMG
    - Identical to compacted plane trees
    - Simpler recurrence relation
    - No asymptotics (yet)
    - Collaborations with Alfredo Viola (Montevideo) and Michael Wallner (TU Wien)

Perspectives

- Study the shape of big DOAGs
- Do something similar for labelled DAGs
  - Collaboration with Philippe Marchal
- Multigraph equivalent: DOAMG
  - Identical to compacted plane trees
  - Simpler recurrence relation
  - No asymptotics (yet)
  - Collaborations with Alfredo Viola (Montevideo) and Michael Wallner (TU Wien)
- What about sparse DOAGs?



- New model: DOAG
- New way of counting
- Control over the number of edges
- When forgetting edges:
  - (Fun?) asymptotic results
  - Optimal uniform sampler

https://github.com/Kerl13/randdag
https://wkerl.me
martin.pepin@lipn.univ-paris13.fr

Outline of the presentation

Background

Directed ordered acyclic graphs

↳ *definition and recursive decomposition*

Asymptotic analysis

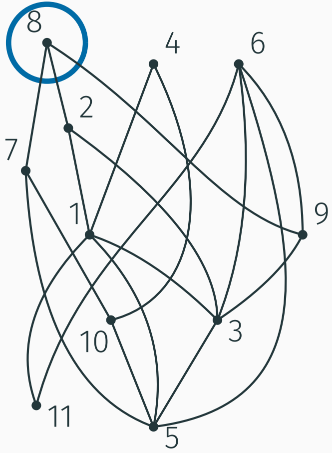
↳ *matrix encoding*
↳ *asymptotic result*
↳ *faster sampler*

Bonus: labelled DAGs

↳ *another way of counting*

What about labelled DAGs?

Idea: mark one source, and remove it.

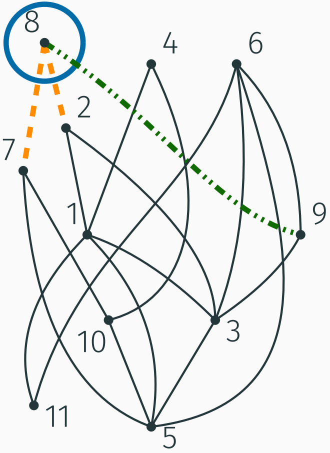


$A_{n,m,k} = \#\text{DAGs}\ (n \text{ vertices}, m \text{ edges}, k \text{ sources})$

$k\, A_{n,m,k} =$

What about labelled DAGs?

Idea: mark one source, and remove it.

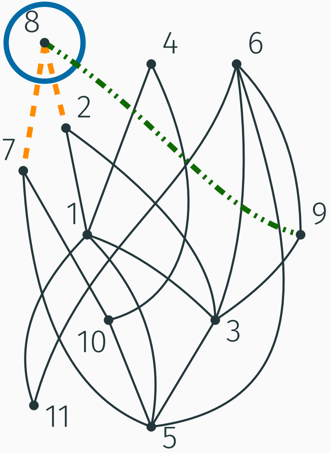


$$A_{n,m,k} = \#\text{DAGs}\,(n \text{ vertices, } m \text{ edges, } k \text{ sources})$$

$$k\,A_{n,m,k} = n \sum_{i,s} A_{n-1,m-i-s,k+s-1} \binom{k+s-1}{s} \binom{n-s-k}{i}$$

What about labelled DAGs?

Idea: mark one source, and remove it.



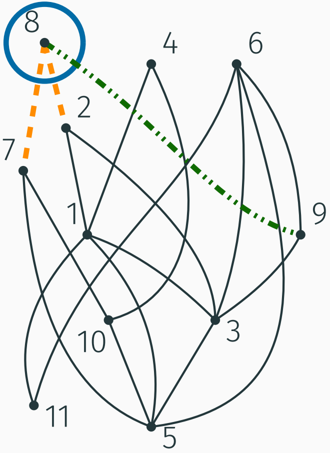
$A_{n,m,k} = \#\text{DAGs}\,(n \text{ vertices, } m \text{ edges, } k \text{ sources})$

$$k\,A_{n,m,k} = n \sum_{i,s} A_{n-1,m-i-s,k+s-1} \binom{k+s-1}{s} \binom{n-s-k}{i}$$

→ New counting formula for DAGs;

What about labelled DAGs?

Idea: mark one source, and remove it.



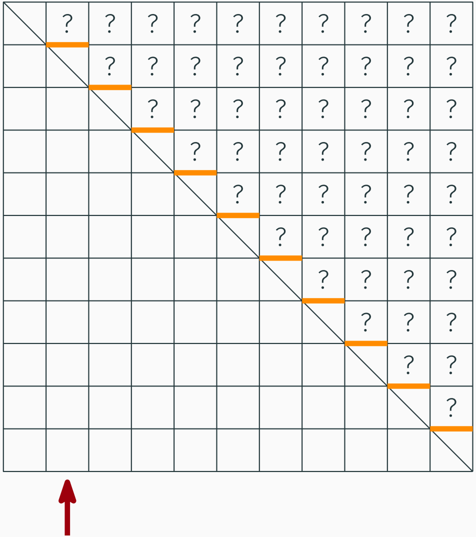
$A_{n,m,k} = \#\text{DAGs}\ (n \text{ vertices}, m \text{ edges}, k \text{ sources})$

$$k\, A_{n,m,k} = n \sum_{i,s} A_{n-1,m-i-s,k+s-1} \binom{k+s-1}{s} \binom{n-s-k}{i}$$

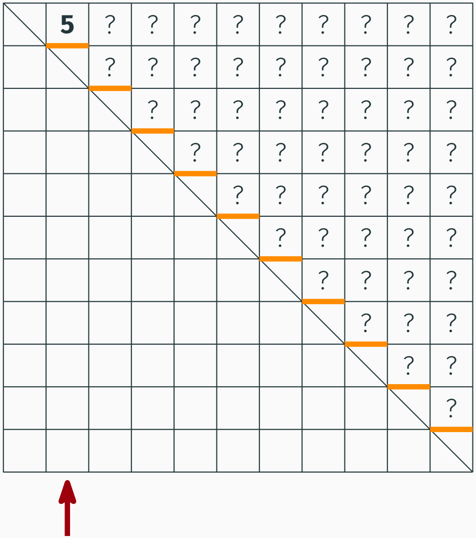
→ New counting formula for DAGs;

→ Effective sampler with fixed number of edges and vertices.

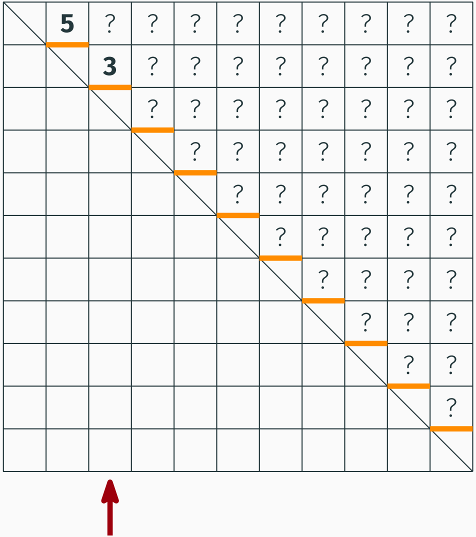
Anticipated rejection



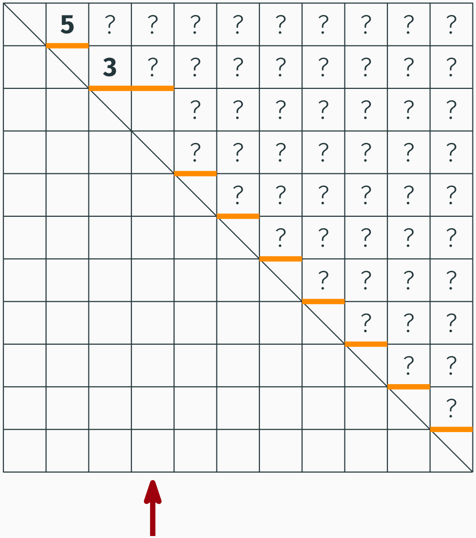
Anticipated rejection



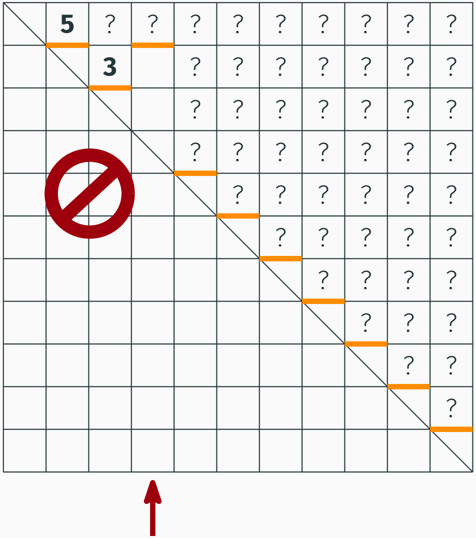
Anticipated rejection



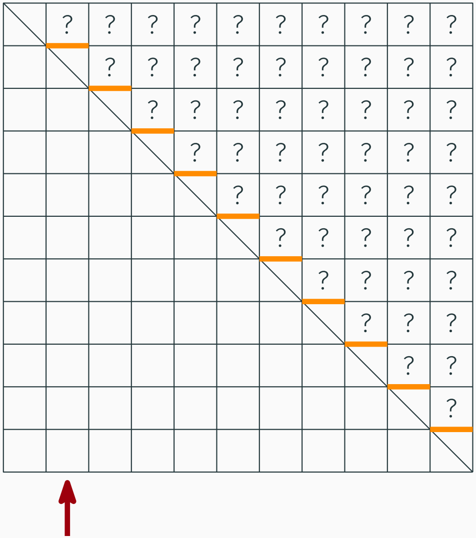
Anticipated rejection



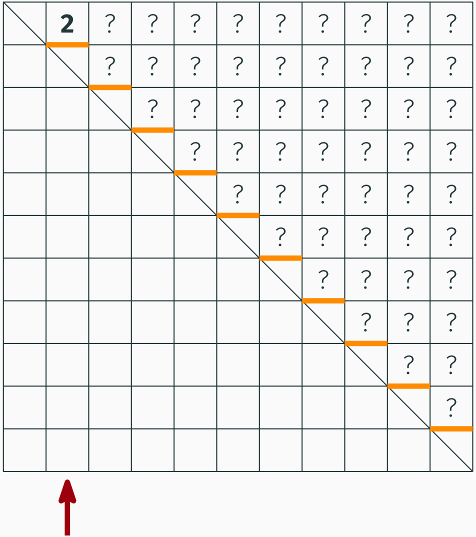
Anticipated rejection



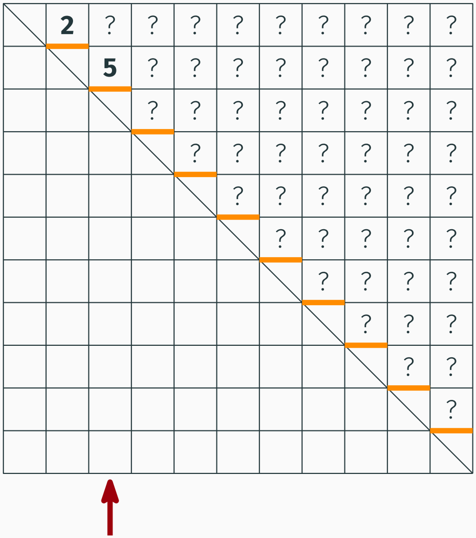
Anticipated rejection



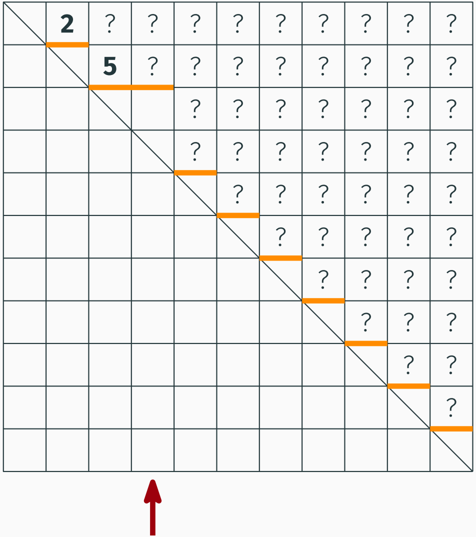
Anticipated rejection



Anticipated rejection

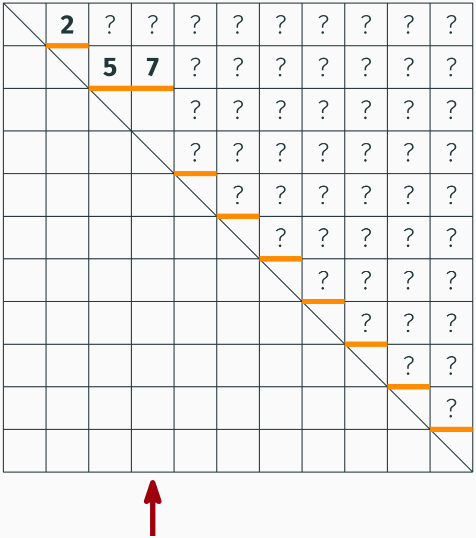


Anticipated rejection



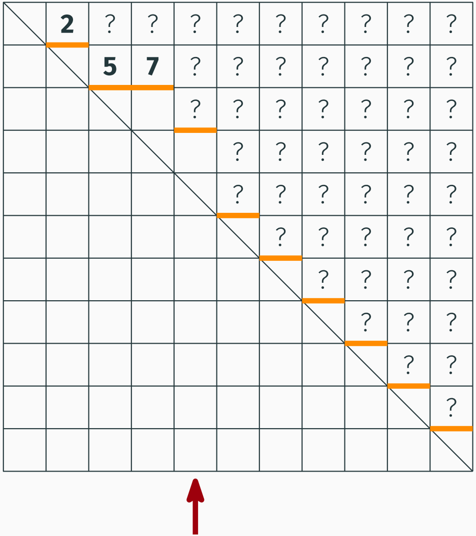
*1

Anticipated rejection

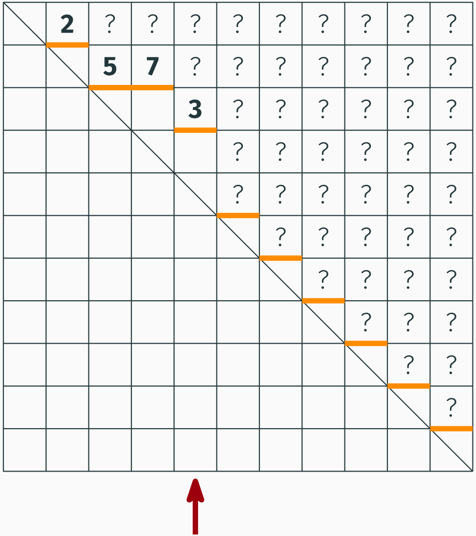


*1

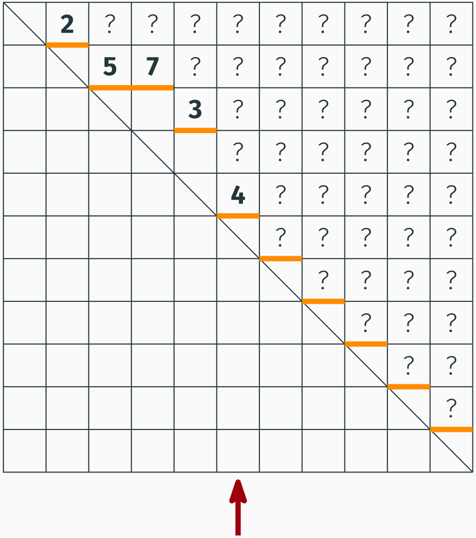
Anticipated rejection



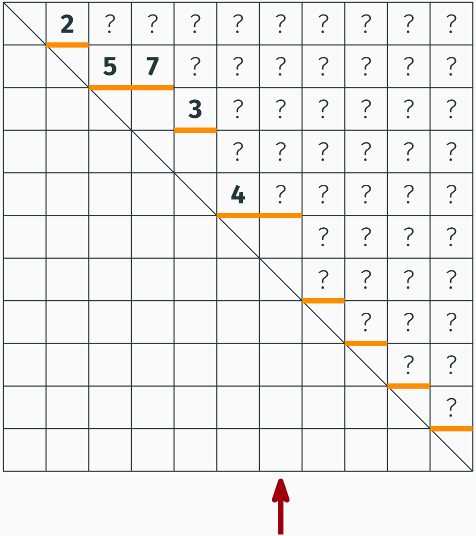
Anticipated rejection



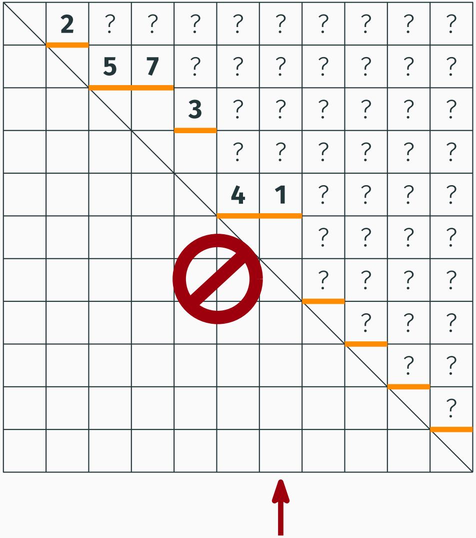
Anticipated rejection



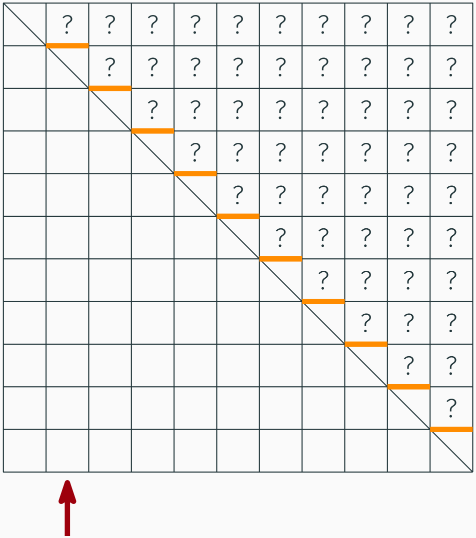
Anticipated rejection



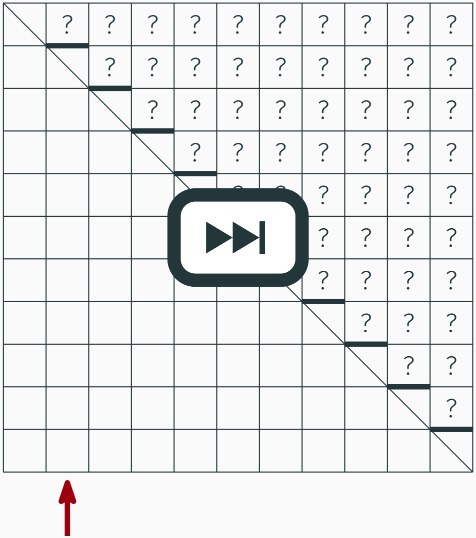
Anticipated rejection



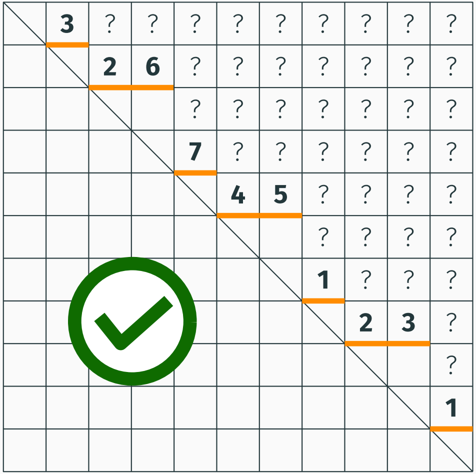
Anticipated rejection



Anticipated rejection

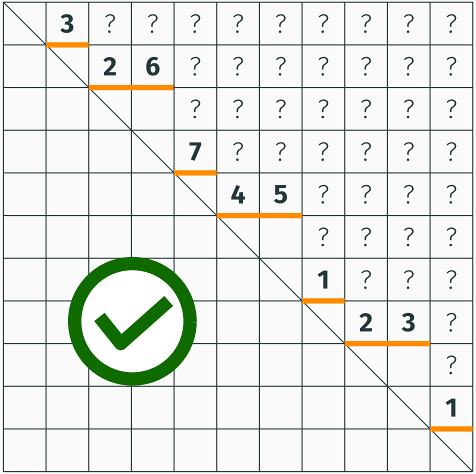


Anticipated rejection



*1

Anticipated rejection



Complexity $= O(n \ln(n))$

Total complexity $= \frac{n^2}{2} \log_2(n) + O\left(\sqrt{n} \cdot n \ln(n)\right)$

*1

References I

[EFW21] Andrew Elvey Price, Wenjie Fang, and Michael Wallner. “Compacted binary trees admit a stretched exponential”. In: *Journal of Combinatorial Theory, Series A* 177 (2021), page 105306. ISSN: 0097-3165. DOI: 10.1016/j.jcta.2020.105306.

[Ges96] Ira Martin Gessel. “Counting acyclic digraphs by sources and sinks”. In: *Discrete Mathematics* 160.1 (1996), pages 253–258. ISSN: 0012-365X.

[GGKW20] Antoine Genitrini et al. “Asymptotic enumeration of compacted binary trees of bounded right height”. In: *Journal of Combinatorial Theory, Series A* 172 (2020), page 105177. ISSN: 0097-3165. DOI: https://doi.org/10.1016/j.jcta.2019.105177.

[KM15] Jack Kuipers and Giusi Moffa. “Uniform random generation of large acyclic digraphs”. In: *Statistics and Computing* 25.2 (2015), pages 227–242.

[MDB01] Guy Melançon, Isabelle Dutour, and Mireille Bousquet-Mélou. “Random Generation of Directed Acyclic Graphs”. In: *Electronic Notes in Discrete Mathematics* 10 (2001), pages 202–207. DOI: 10.1016/S1571-0653(04)00394-4. URL: https://doi.org/10.1016/S1571-0653(04)00394-4.

References II

[Rob70] Robert William Robinson. “Enumeration of acyclic digraphs”. In: *Proceedings of The Second Chapel Hill Conference on Combinatorial Mathematics and its Applications (Univ. North Carolina, Chapel Hill, NC, 1970), Univ. North Carolina, Chapel Hill, NC* (University of North Carolina at Chapel Hill, North Carolina, May 8–13, 1970). 1970, pages 391–399.

[Rob73] Robert William Robinson. “Counting labeled acyclic digraphs”. In: *New Directions in the Theory of Graphs* (1973), pages 239–273.

[Rob77] Robert William Robinson. “Counting unlabeled acyclic digraphs”. In: *Combinatorial Mathematics V*. Lecture Notes in Mathematics. Springer, 1977, pages 28–43.

[Sta73] Richard Peter Stanley. “Acyclic orientations of graphs”. In: *Discrete Mathematics* 5.2 (1973), pages 171–178.